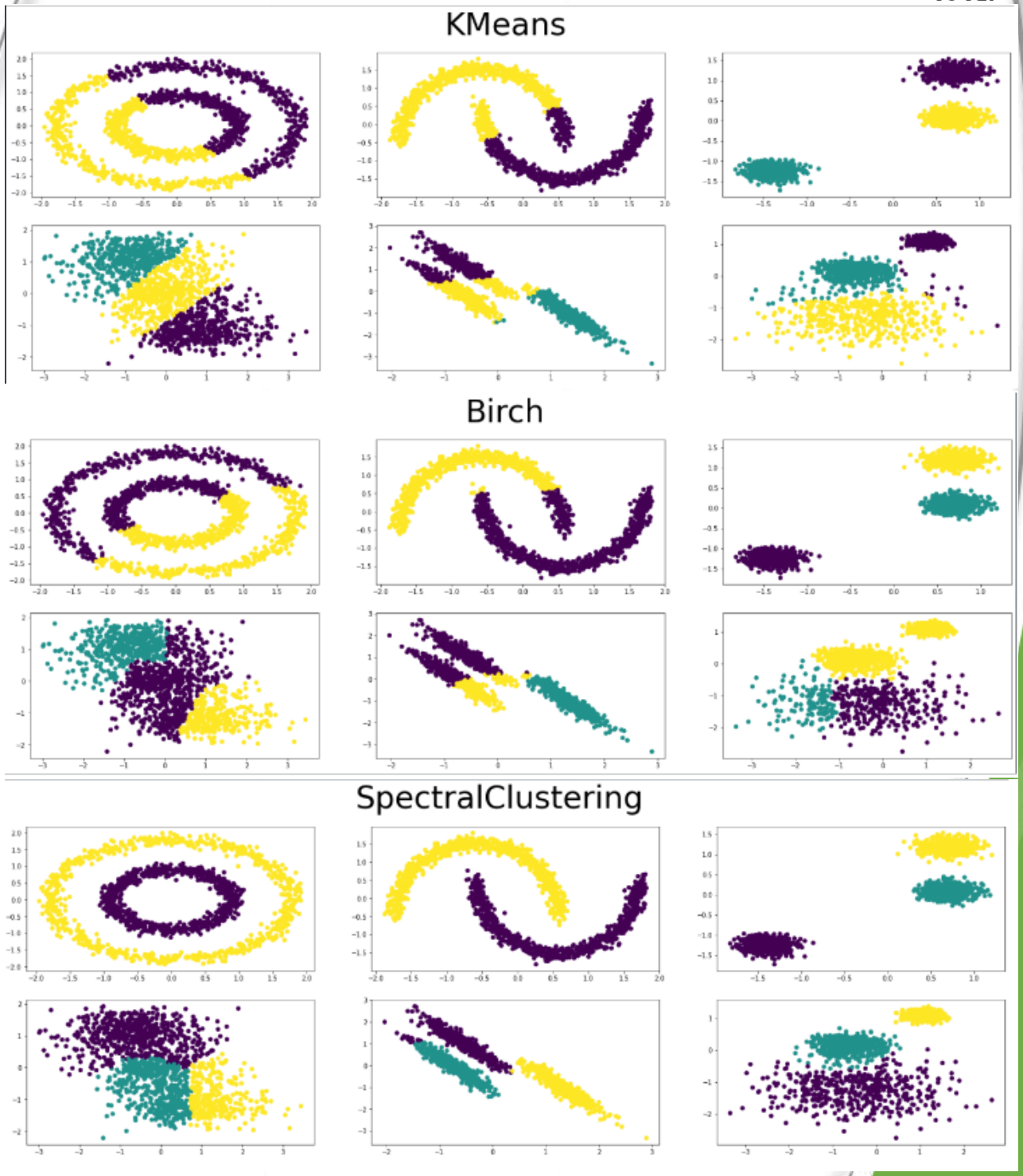
Actividad 10 Comparacion de Algoritmos de Agrupacion

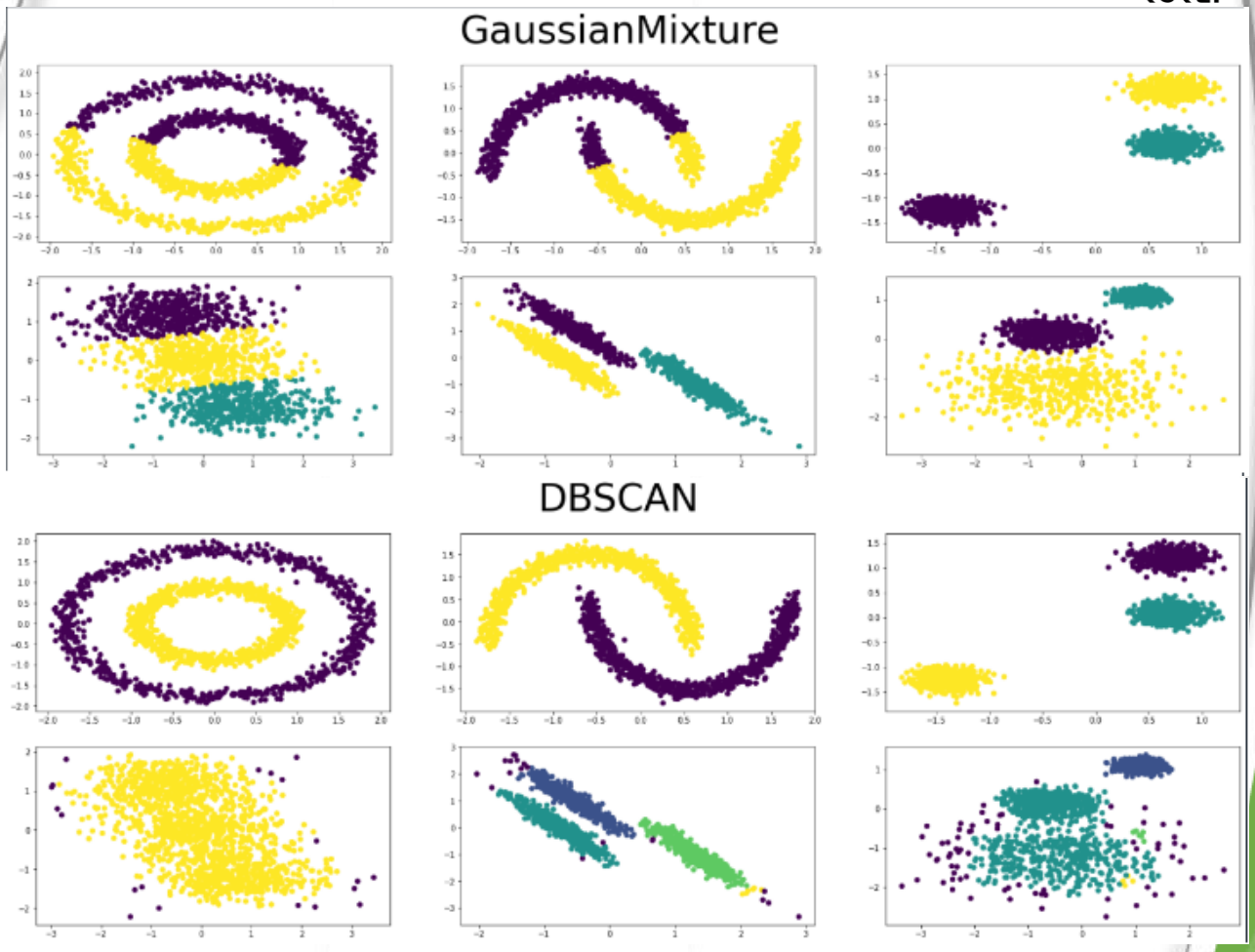
Alumno: José Osvaldo Farfán de León

Materia: Seminario de IA ll

Profesor: Carlos Alberto Villaseñor Padilla

Sección: “D04”


KMeans
Birch
SpectralClustering


GaussianMixture
DBSCAN

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import cluster,datasets,mixture,metrics
from sklearn.preprocessing import StandardScaler

#creacion de datasets-----------------------
np.random.seed(0)
n_samples = 1500
X = 6 * [None]

#circulos concentricos
xtemp, _ = datasets.make_circles(n_samples=n_samples, factor=0.5, noise=0.05)
X[0] = StandardScaler().fit_transform(xtemp)

#lunas
xtemp, _ = datasets.make_moons(n_samples=n_samples, noise=0.05)
X[1] = StandardScaler().fit_transform(xtemp)

#Blobs
xtemp, _ = datasets.make_blobs(n_samples, random_state=8)
X[2] = StandardScaler().fit_transform(xtemp)

#PLano sin agrupacion
xtemp, _ = datasets.make_blobs(n_samples, 2)
X[3] = StandardScaler().fit_transform(xtemp)

#Blobs con deformacion anisotropica
xtemp, _ = datasets.make_blobs(n_samples, random_state=170)
xtemp = np.dot(xtemp, [[0.6,-0.6],[-0.4,0.8]])
X[4] = StandardScaler().fit_transform(xtemp)

#blobs con diferentes varianzas
xtemp, _ =datasets.make_blobs(n_samples = n_samples, random_state = 43,
                              cluster_std = [1.0, 2.5, 0.5])
X[5] =StandardScaler().fit_transform(xtemp)
n_cluster = [2,2,3,3,3,3]

y = []
for c, x in zip(n_cluster, X):
    model = cluster.KMeans(n_clusters=c)
    model.fit(x)
    if hasattr(model, 'labels_'):
        y.append(model.labels_.astype(np.int))
    else:
        y.append(model.predict(x))
#-----------------------------------------------
plt.figure(figsize=(27,9))
plt.suptitle('KMeans', fontsize = 48)
for i in range(6):
    ax = plt.subplot(2, 3,i+1)
    ax.scatter(X[i][:,0], X[i][:,1],c=y[i])
```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import cluster,datasets,mixture,metrics
from sklearn.preprocessing import StandardScaler

#creacion de datasets-----------------------
np.random.seed(0)
n_samples = 1500
X = 6 * [None]

#circulos concentricos
xtemp, _ = datasets.make_circles(n_samples=n_samples, factor=0.5, noise=0.05)
X[0] = StandardScaler().fit_transform(xtemp)

#lunas
xtemp, _ = datasets.make_moons(n_samples=n_samples, noise=0.05)
X[1] = StandardScaler().fit_transform(xtemp)

#Blobs
xtemp, _ = datasets.make_blobs(n_samples, random_state=8)
X[2] = StandardScaler().fit_transform(xtemp)

#PLano sin agrupacion
xtemp, _ = datasets.make_blobs(n_samples, 2)
X[3] = StandardScaler().fit_transform(xtemp)

#Blobs con deformacion anisotropica
xtemp, _ = datasets.make_blobs(n_samples, random_state=170)
xtemp = np.dot(xtemp, [[0.6,-0.6],[-0.4,0.8]])
X[4] = StandardScaler().fit_transform(xtemp)

#blobs con diferentes varianzas
xtemp, _ =datasets.make_blobs(n_samples = n_samples, random_state = 43,
                              cluster_std = [1.0, 2.5, 0.5])
X[5] =StandardScaler().fit_transform(xtemp)

n_cluster = [2,2,3,3,3,3]

y = []
eps = [0.3,0.3,0.3,0.3,0.15,0.18]
for e, x in zip(eps, X):
    model = cluster.DBSCAN(eps=e)
    #model = cluster.SpectralClustering(n_clusters=c, affinity='nearest_neighbors'
    #model = cluster.KMeans(n_clusters=c)
    #model = cluster.Birch(n_clusters=c)
    model.fit(x)
    if hasattr(model, 'labels_'):
        y.append(model.labels_.astype(np.int))
    else:
        y.append(model.predict(x))
#-------------------------------------------------
plt.figure(figsize=(27,9))
plt.suptitle('DBSCAN', fontsize = 48)
#plt.suptitle('KMeans', fontsize = 48)
for i in range(6):
    ax = plt.subplot(2, 3,i+1)
    ax.scatter(X[i][:,0], X[i][:,1],c=y[i])
```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import cluster,datasets,mixture,metrics
from sklearn.preprocessing import StandardScaler

#creacion de datasets-----------------------
np.random.seed(0)
n_samples = 1500
X = 6 * [None]

#circulos concentricos
xtemp, _ = datasets.make_circles(n_samples=n_samples, factor=0.5, noise=0.05)
X[0] = StandardScaler().fit_transform(xtemp)

#lunas
xtemp, _ = datasets.make_moons(n_samples=n_samples, noise=0.05)
X[1] = StandardScaler().fit_transform(xtemp)

#Blobs
xtemp, _ = datasets.make_blobs(n_samples, random_state=8)
X[2] = StandardScaler().fit_transform(xtemp)

#PLano sin agrupacion
xtemp, _ = datasets.make_blobs(n_samples, 2)
X[3] = StandardScaler().fit_transform(xtemp)

#Blobs con deformacion anisotropica
xtemp, _ = datasets.make_blobs(n_samples, random_state=170)
xtemp = np.dot(xtemp, [[0.6,-0.6],[-0.4,0.8]])
X[4] = StandardScaler().fit_transform(xtemp)

#blobs con diferentes varianzas
xtemp, _ =datasets.make_blobs(n_samples = n_samples, random_state = 43,
                              cluster_std = [1.0, 2.5, 0.5])
X[5] =StandardScaler().fit_transform(xtemp)

n_cluster = [2,2,3,3,3,3]

y = []
for c, x in zip(n_cluster, X):

    model = mixture.GaussianMixture(n_components=c,covariance_type='full')
    #model = cluster.SpectralClustering(n_clusters=c, affinity='nearest_neighbors')
    #model = cluster.KMeans(n_clusters=c)
    #model = cluster.Birch(n_clusters=c)
    model.fit(x)
    if hasattr(model, 'labels_'):
        y.append(model.labels_.astype(np.int))
    else:
        y.append(model.predict(x))
#--------------------------------------------------
plt.figure(figsize=(27,9))
plt.suptitle('GaussianMixture', fontsize = 48)
#plt.suptitle('KMeans', fontsize = 48)
for i in range(6):
    ax = plt.subplot(2, 3,i+1)
    ax.scatter(X[i][:,0], X[i][:,1],c=y[i])
```